University of St Andrews

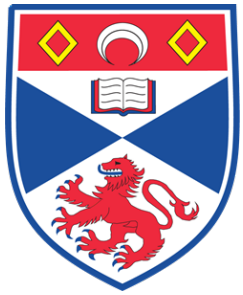
CS5014

Practical 1

Machine Learning

Author:
150008022

March 8, 2019



Goal

The goal of this practical was to cleanse and process real world data in order to produce a regression model, and evaluate its performance.

Contents

Part I
Loading and Cleaning the Data

1 Loading

Numpy was used to load the csv file. The header row was skipped, and to ensure there were no missing values, the invalid raise flag was also used when parsing the data. This would raise an exception if any rows were found to be missing data.

No preproccessing had occurred on the data set according to the original paper, and so the data could be used as given.

The input and output columns were separated into two variables x and y, in accordance with the notation used in lectures.

2 Data Splitting

Next, the testing set was isolated from the available data. Simple random sampling (SRS) and stratified sampling were considered for performing the data splitting. SRS is intuitive, but for less uniformly distributed data sets, it can lead to subsets that poorly represent the input data, and therefore suffer from sampling bias [1].

For these reasons, stratified random sampling was used with a 80%-20% split. To ensure our training and testing sets were representative of our data, the output variables were used to categorise the data into strata. Since the two appeared to have a linear relationship as shown in figure 5, we could assume that using one column for categorising our data would produce an roughly equal distribution of the values in both columns.

The 586 possible values of Y1 were split into 50 bins using `numpy.digitize`, and then passed to `train_test_split` as the `stratify` argument.

Part II
Analysing and Visualising the Data

1 Distributions

Firstly, a histogram of each of the input variables and outputs were plotted in order to visualise the distribution of the values (Figures 1 and 2). When comparing to the histograms from the given paper [2], most of the plots matched. Any differences were identified to be caused by 10 bins always being used (the default if not specified by numpy.hist), and the paper would sometimes use more. However, it was still clear that none of the variables had a gaussian distribution. The output variables also appeared tail heavy, but the inputs did not.

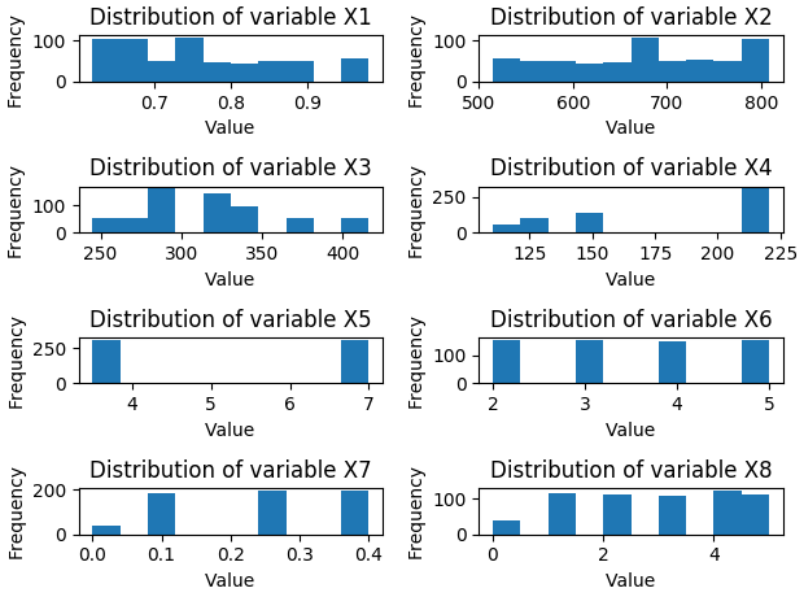


Figure 1: Distribution of input variables

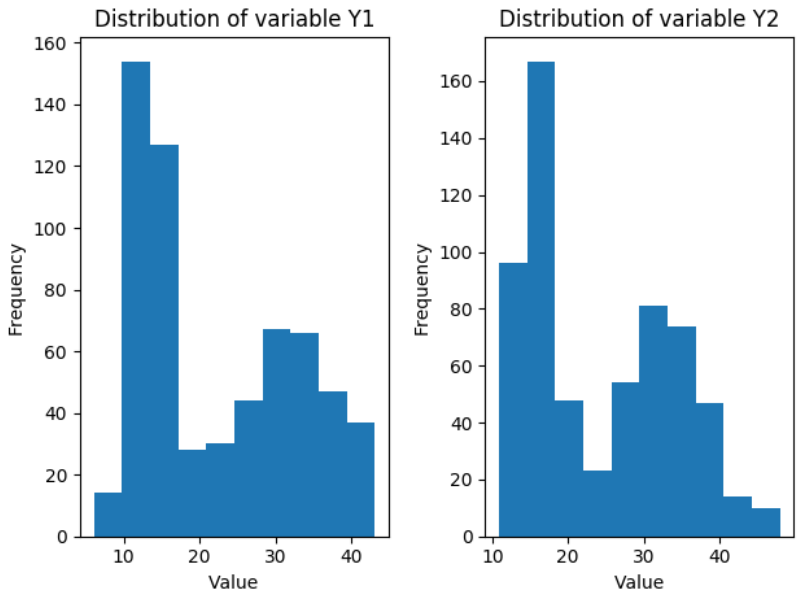


Figure 2: Distribution of output variables

2 Relationships

In order to identify which variables had the strongest relationships with the outputs, and if these relationships were linear or monotonic, scatter graphs were made between each input and output variable (Figures 3 and 4). The inputs were normalised to allow for comparisons between values that could have very different ranges. To better visualise the density of the data points as well as their position, the `alpha` parameter was set to 0.1 in the plots.

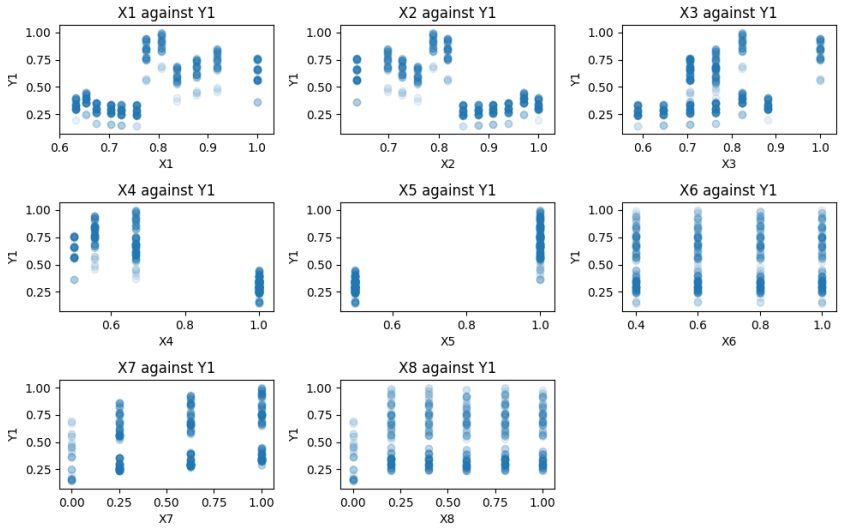


Figure 3: Normalised inputs plotted against the first output variable

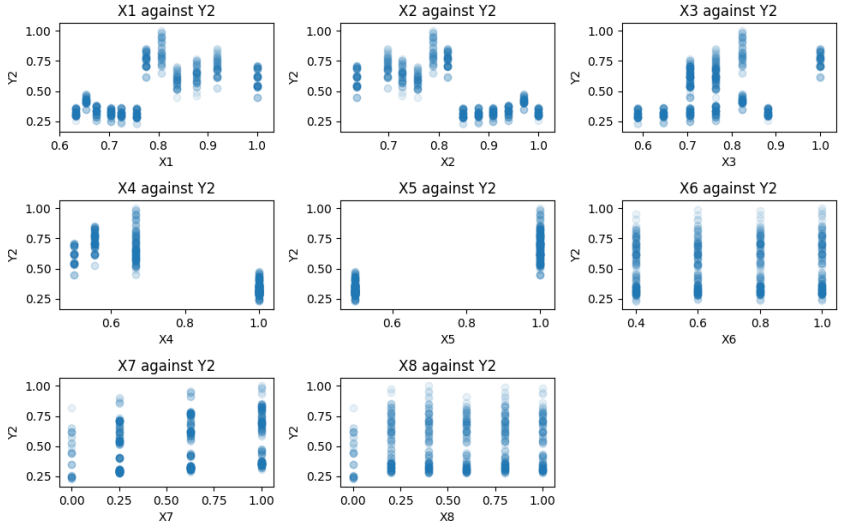


Figure 4: Normalised inputs plotted against the second output variable

Since the distribution of the outputs were so similiar, they were also plotted against each other in a scatter plot (Figure 5). This showed that the two variables had very similiar values, and so a regression model that applied to only one of them could likely be used for the other.

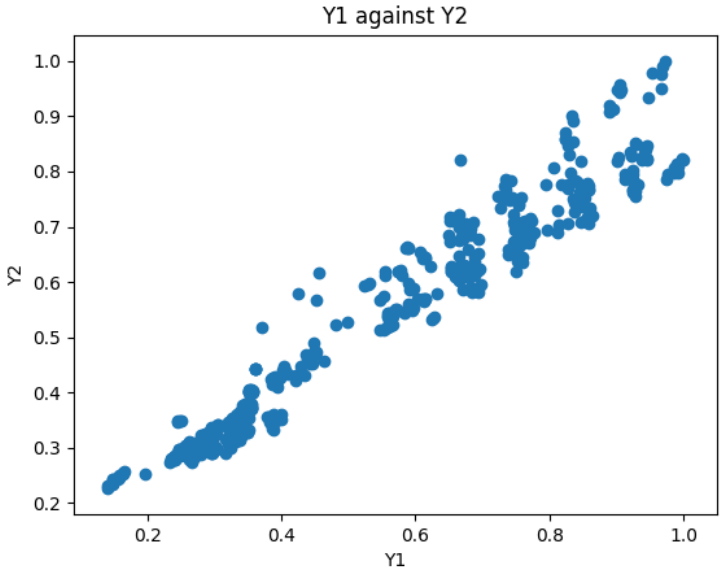


Figure 5: Normalised outputs against each other

Part III
Feature Selection

1 Correlation Ranking

To try and identify which features had the strongest effect on the outputs, both the Pearson and Spearman rank correlation coefficients were considered. It was noted that the Pearson correlation would give a perfect value when the two variables were linearly related, whilst the Spearman correlation (a similiar alternative, and the one used in the original paper) would give a perfect value when the variables were monotonically related. Given these factors, the Spearman rank correlation coefficient was used as a filter method.

From the scatter plots, some variables appeared to have a possible linear relationship with the outputs (for example, X7 and Y1), whilst others had a monotonic relationship (for example, X1 and Y1). Using the Scipy *stats.spearmanr* method, the correlation coefficients were easily calculated, alongside a p-value, as shown in table 1. Immediately from these result we

can see that X6 and X8 show little evidence of a monotonic relationship existing between them and either of the outputs.

However, according to Schiavon et al. [3], the orientation (X6) is one of the most important predictors for cooling load (Y2). Though fewer features would improve the processing time, and removing less important features would be crucial for eliminating noise, only X8 was removed from our feature set.

X	Y	Rho	p
1	1	0.61	0.00
1	2	0.64	0.00
2	1	-0.61	0.00
2	2	-0.64	0.00
3	1	0.49	0.00
3	2	0.43	0.00
4	1	-0.80	0.00
4	2	-0.80	0.00
5	1	0.86	0.00
5	2	0.86	0.00
6	1	0.00	0.99
6	2	0.03	0.51
7	1	0.35	0.00
7	2	0.32	0.00
8	1	0.09	0.03
8	2	0.06	0.14

Table 1: Spearman rank correlation coefficients, with p values

Part IV
Selecting and Training a Regression Model

1 Model Selection

The limited size of the corpus available made the choice of algorithm especially crucial. The lack of noticable outliers in the visualistions at least suggest that the data is not of poor-quality.

Scikit-Learns `DecisionTreeRegressor` [4] which implements CART was used as the regression model due to the complex relationships that appeared to exist between the inputs and outputs, using mean squared error as the cost function.

Linear regression was deemed inappropriate as it is a global model. The relationships between the inputs as well as their individual valuddces seemed to have an effect on the output, and so a model that captured these inter-variable relationships would likely be more effective. The danger with using CART was that it could be more prone to overfitting, but placing limits on the complexity of the decision tree could reduce this risk.

The model is constructed by recursively breaking down the training data and building a decision tree from each subsection of the data. The tree can then be used to make a prediction based on some input x by starting at the root node of the tree and using a series of questions to find the leaf node that would be our prediction. For example, one of the questions might be "is X1 greater than 0.5?" or "is X1 greater than X2?". CART constructs a binary decision tree by using the attribute that provides the most information gain at each stage.

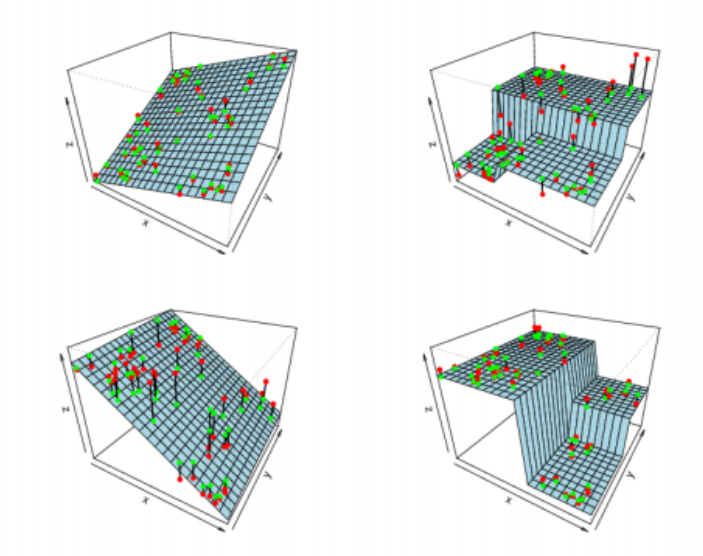


Figure 6: Comparison of CART and Linear Regression [5]

2 Cost Function Selection

Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) were both considered as performance measures for regression. RMSE minimizes for the squared difference between the estimated and expected values, whilst MAE minimizes for the absolute difference, therefore RMSE is more affected by outliers, which can result in overfitting. However the input data did not seem to contain many outliers, and so RMSE was chosen.

3 Validation Method

Due to the relatively small corpus size, K-fold cross-validation was chosen to estimate the model error [6]. 10 folds were used as it provides a good compromise between bias and variance [7].

4 Optimisations

By assessing the accuracy of the CART model at limited depths, we could attempt to limit the complexity of the decision tree created, and therefore reduce the chance of overfitting. By iteratively using k-fold cross validation and RMSE to produce an accuracy score at increasing depths, we would be able to find the point where increasing complexity does not benefit the model [8].

Figure 7 shows that performance begins to deteriorate once depth is greater than 6. Accuracy alone is not enough to confirm a models effeccd tiveness, but can build confidence in the model when coupled with another method such as cross-validation.

Figure 7: Accuracy of model with increasing decision tree depth

5 Other Observations

Out of curiosity, the 'importance' of each variable was also plotted to better understand what the regression model would be most affected by, and to see if this aligned with the feature selection methods used earlier.

Figure 8 shows that the two models are strongly dependent on different variables. Relative compactness has the strongest effect on the heating load, whilst overall height has the strongest effect on the cooling load.

This is not the expected result based on the spearman rank correlation coefficients calculated in table 1. There also appears to be very little dependence in both models on X6 despite the results referred to from Schiavon et al [3].

When the `random_state` parameter was adjusted for the model, the distribution of feature importances remained roughly the same, and X1 and X5 were consistently the most important features. However, the performance of the model (discussed later) did not depend on this value. The parameter provides a seed for a random number generator that can allow the results to be reproduced, but a model that can generalise should not depend on this value.

(a) Feature Importance for Y1 (b) Feature Importance for Y2

Figure 8: Feature Importance for models trained on the different outputs

The feature importance calculation used by default in `sklearn` is Mean Decrease Impurity (MDI), where the values are calculated based on how many times a variable is used to split the decision tree. Another method that can be used is permutation importance (PI), which instead shuffles the values for an attribute and measures how much this affects the model. If the output is heavily dependent on the shuffled attribute, then the accuracy should drop sharply when the values are shuffled.

In comparison, MDI is quicker as it doesn't explore several permutations of the attributes. On the other hand, since numerical variables have more

split points than categorical data, MDI can be biased towards input variables of that type [9].

The `eli5.sklearn` module was used to perform the permutation importance, which had very interesting results, shown in tables 2 and 3. It still showed X5 as the most importance for Y1 and X1 for Y2, but this was consistent regardless of the random state parameter.

Table 2: PI for Y1 model

Variable	Importance	Error (±)
X1	0.30	0.03
X2	0.02	0.00
X3	0.02	0.01
X4	0.00	0.00
X5	2.49	0.36
X6	-0.00	0.00
X7	0.22	0.04

Table 3: PI for Y2 model

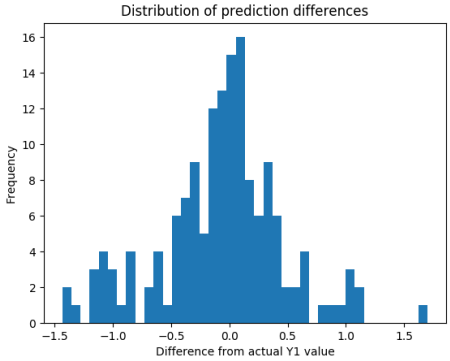
Variable	Importance	Error (±)
X1	2.61	0.14
X2	0.34	0.11
X3	0.06	0.01
X4	0.00	0.00
X5	0.00	0.00
X6	-0.00	0.00
X7	0.12	0.03

Feature importance calculations using these methods can be used for feature selection, but wasn't for this practical.

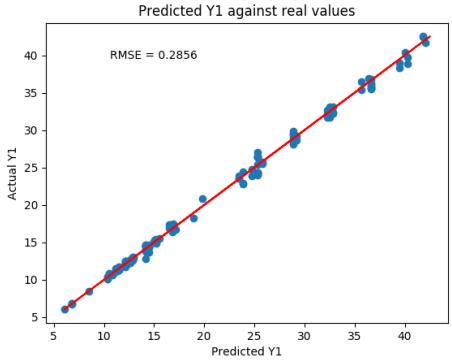
Part V
Performance Evaluation

1 Results

The model was trained on the training data with the max depth set to 6, and then used to predict the outputs using the test data isolated earlier. A seperate model was used for predicting each of the output variables. The differences between the predicted values and actual values are visualised in figures 9 and 10.

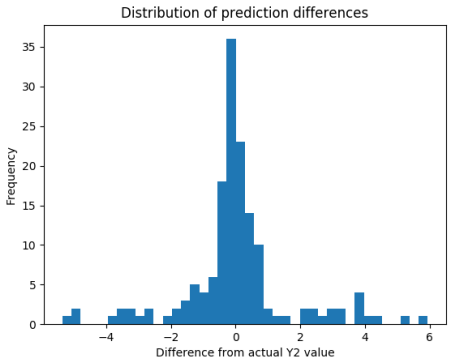


(a) Distribution of differences for Y1

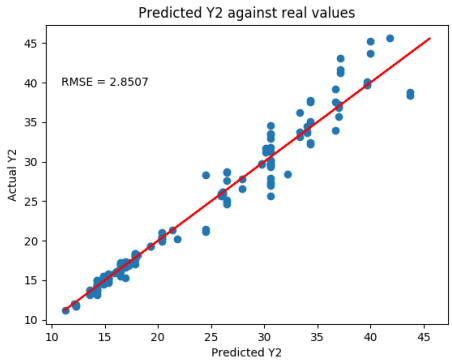


(b) Prediction against Actual for Y1

Figure 9: Error visualisation for Y1



(a) Distribution of differences for Y2



(b) Prediction against Actual for Y2

Figure 10: Error visualisation for Y

2 Evaluation

The model seems to perform well with both outputs, and especially with Y1, based solely on the RMSE values. It performed just as well on the training data as it did on the test data, which suggests that overfitting has not occurred. For Y2, the RMSE is 0.08% of the range of values, which should be accurate enough for most simulations that it could be used for.

The feature importance calculations disagree with the results from Tsanas and Xifara [2], where they found X7 to be the best predictor. Similiar error

was obtained in the predictions for Y1 , but not for Y2.

Part VI
Discussion

Due to the way the input data was generated, it is possible that the model could not be used globally. For example, the building simulation used Greece as a location, which has very different conditions to that of Alaska, for example.

Part VII
Conclusion

The practical proved very interesting, and comparing our interpretation of the data with that of an actual research paper was motivating. For each component in the practical, alternatives were considered and discussed in order to expand my knowledge of what tools were available and when each of them were most applicable.

References

[1] Z. Reitermanova. *Data Splitting*. Matfyzpress, 2010.

[2] Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

[3] Stefano Schiavon, Kwang Ho Lee, Fred Bauman, and Tom Webster. Influence of raised floor on zone design cooling load in commercial buildings. *Energy and Buildings*, 42(8):1182–1191, 2010.

[4] Scikit-learn decision trees. `https://scikit-learn.org/stable/modules/tree.html#tree`, cited March 2019.

[5] Leo Breiman. Regression trees. *Classification And Regression Trees*, page 216–265.

[6] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow*. O'Reilly Media, 2018.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[8] Luca Massaron and John Mueller. *Python for Data Science for Dummies*. John Wiley & Sons, Inc., 2015.

[9] Terence Parr, Kerem Turgutlu, Christopher Csiszar, and Jeremy Howard. Beware default random forest importances. `https://explained.ai/rf-importance/index.html`, cited March 2019.